山东大学
SHANDONG UNIVERSITY

崇新学堂

2024 - 2025 学年第一学期

实验报告

课程名称：Introduction to EECS Lab

实验名称：Design Lab4 - Hitting the Wall

学生姓名：胡君安、陈焕斌、黄颢

实验时间：2024 年 10 月 31 日

1 Goals

The overall goal of this lab is to use a simple proportional controller, and model it as a system with the same structure that we used for the wall finder system of Design Lab 4 to program the robot to move along a wall to its side, maintaining a constant, desired distance from the wall. We will achieve this in three stages:

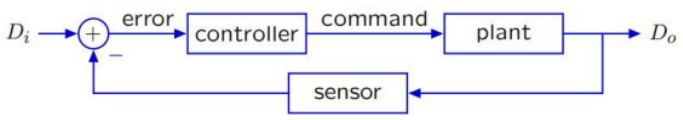
- Build a proportional controller for a robot and test it in simulation with different gains.
- Build an analytical model of the controller-plant-sensor system, both by hand and using the SystemFunction class.
- Use the model to gain understanding about the best gain to use for the controller and what kind of behavior to expect from the system.

2 Experimental step

Collect resources

- **propWallFollowBrainSkeleton.py:** Template brain with a place for you to write the proportional controller.
- **designLab05Work.py:**Template python script with appropriate imports for making system functions and finding their properties.
- **Chapter5** of the course notes.

This week, we generalize from the simple one-dimensional model, to a two-dimensional world,and seek to guide a robot such that it moves parallel to a wall, staying a specific distance away. In addition, we generalize from difference equations to a model using abstract signals, and system equations.



Check Yourself 1

What should be the types of the input to and the output from a state machine of the WallFollower class? What should be the sign of k?

Answer : The input is the distance from the car to the wall. K should be a negative number.

Checkoff 1

Show your plots to a staff member. Describe how the gain k affects behavior of the wall-follower. For what value of k (if any) does the distance to the wall converge most rapidly to the desired value of 0.5 m?

Compare the effect of k in this lab and in the lab 4 (wall finder).

Step 1

We implemented the proportional controller by editing the brain. Then use soar to run brain in the world wallTestWorld.py.

The code is as follows:

```
class WallFollower(sm.SM):
    k=-80
    T=0.1
    def getNextValues(self,state,inp):
        AngleVariation=self.k*(inp-desiredRight)* self.T
        if inp >= desiredRight:
            if -0.1< AngleVariation< 0.1:
                return state,io.Action(fvel=forwardVelocity,rvel=0)
            else:
                return state, io.Action(fvel=forwardVelocity,rvel=(inp - desiredRight)
                                        * self.k)
        else:
            return state, io.Action(fvel=forwardVelocity,rvel=(inp - desiredRight)
                                    * self.k)
```

Step 2

We completed the procedure and used three k values, k=-1, k=20, k=-20, k=-80, k=-100, k=-200 to test whether our simulation works correctly. Below is our program and images of the results of our experiment.

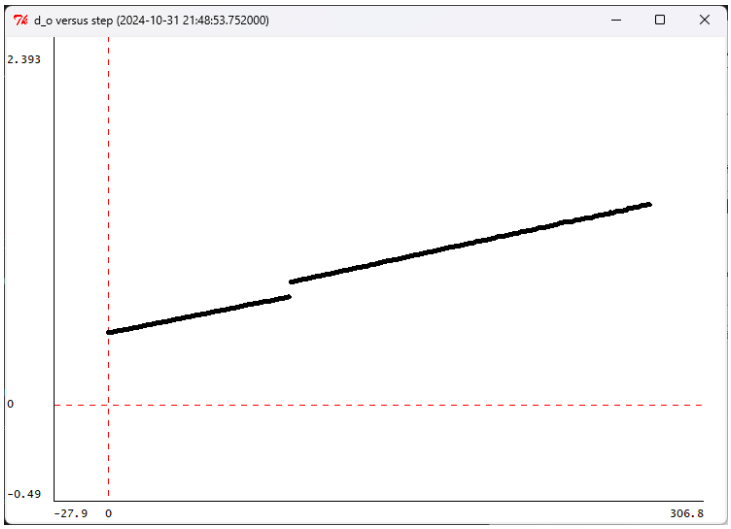


Fig. 1. k = -1

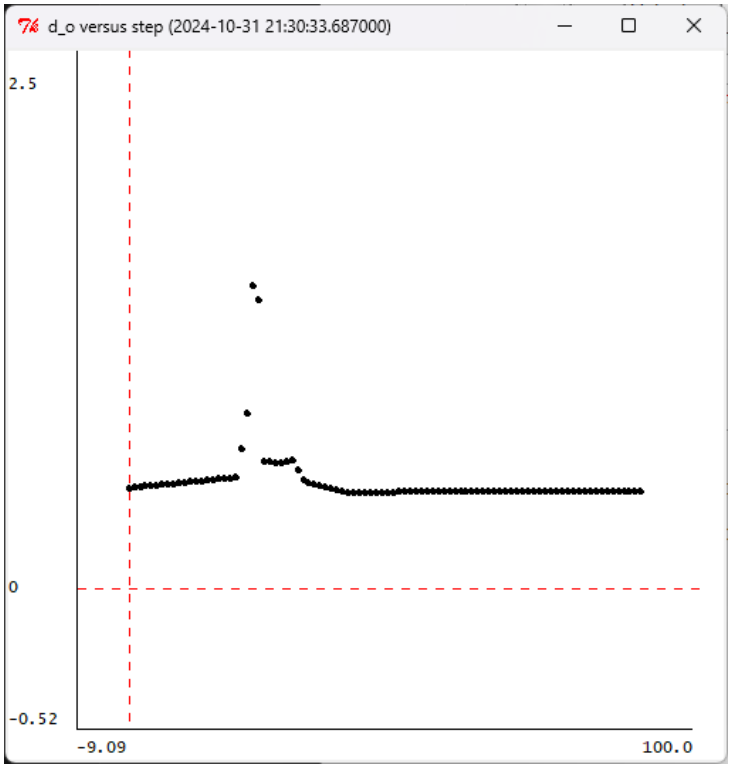


Fig. 2. k = 20

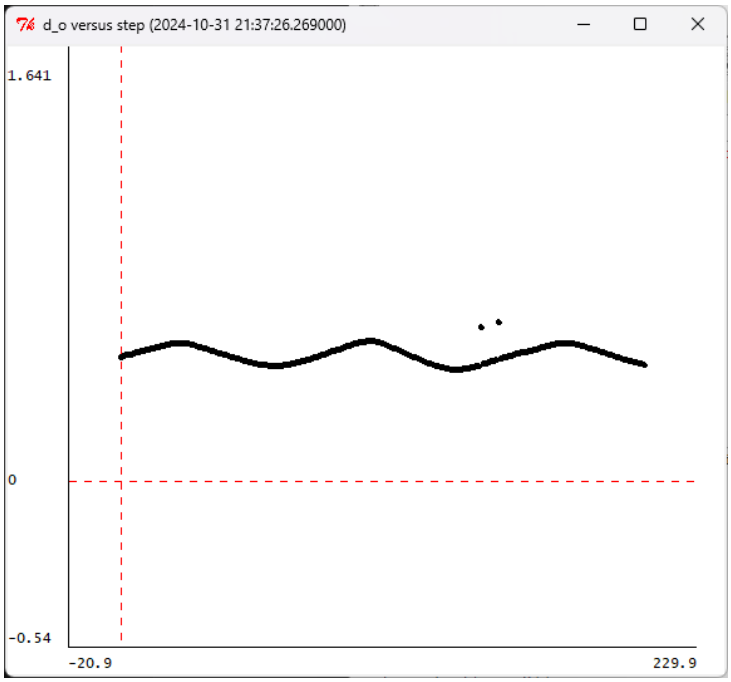


Fig. 3. k = -20

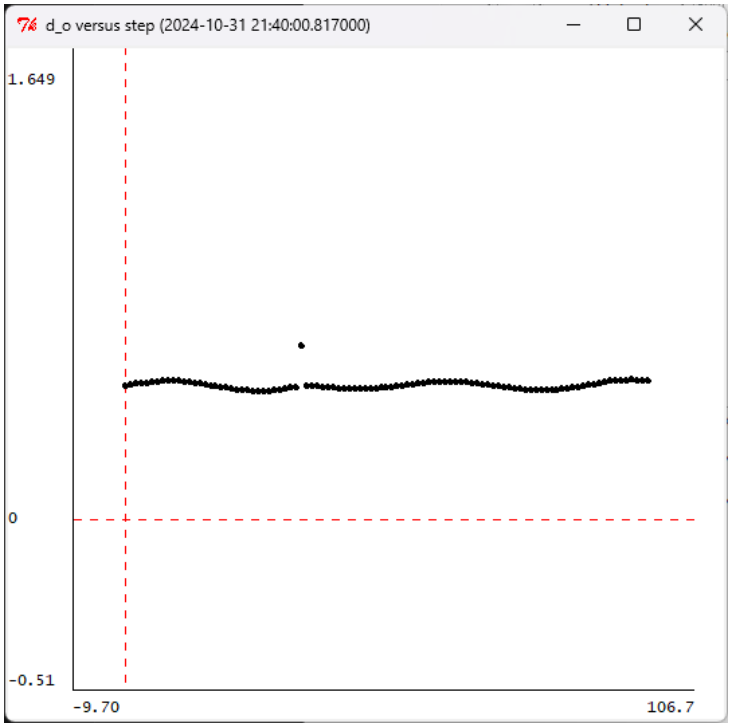


Fig. 4. k = -80

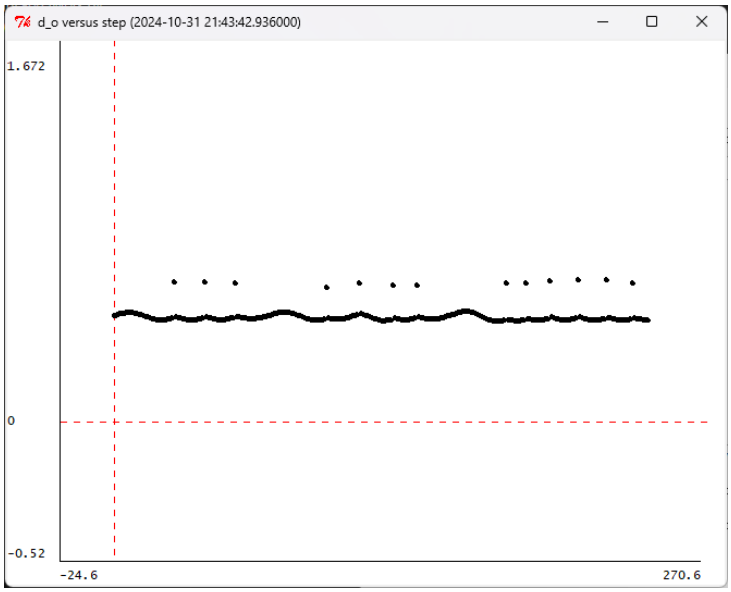


Fig. 5. k = -100

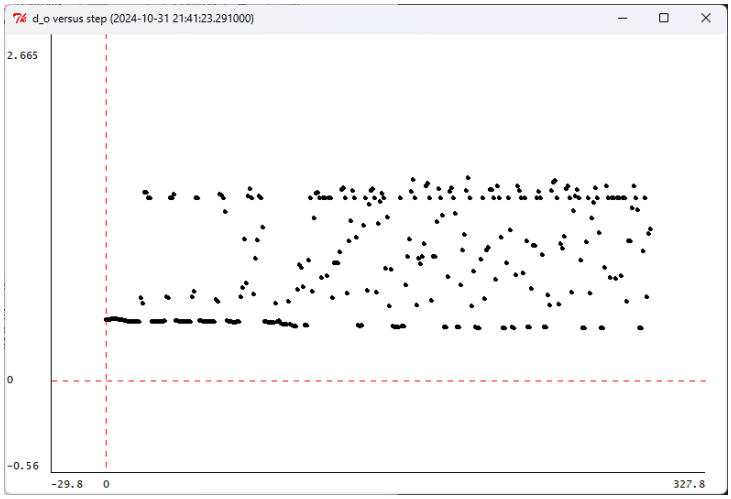


Fig. 6. k = -200

It' s not difficult to find that the larger the absolute value of k is, the more sensitive the car's perception of the wall is. When k is between -30 and -80, it is more suitable. Compared to Design Lab 4, k is no longer a monotonic function.

Step 3

Controller model: Assume that the controller can instantly set the rotational velocity $\omega[n]$ to be proportional to the error $e[n]$. Express this relation as a difference equation.

The expression for Controller:

$$\omega[n] = k(d_i[n] - d_o[n])$$

Step 4

Write difference equations describing the "plant," which relates the input (angular velocity) to the output (distance to the wall).

The expression for Plant 1:

$$\theta[n] = \theta[n-1] + T\omega[n-1]$$

The expression for Plant 2:

$$d_o[n] = d_o[n-1] + TV\theta[n-1]$$

Sensor model: The subsystems represented by the three difference equations above connect together to form a system of the following form.

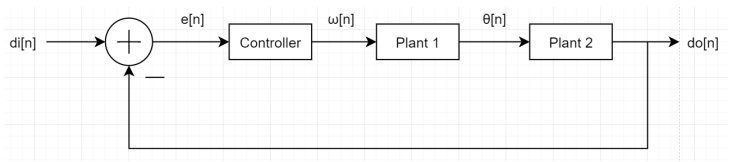


Fig. 7. Sensor model

Step 5

Convert your difference equations into operator(R) equations, and find the system function.

$$H = \frac{-T^2VKR^2}{(1+T^2VK)R^2 - 2R + 1}$$

Step 6

Find an algebraic expression for the poles p of the system.

$$P = 1 \pm \sqrt{T^2VK} \cdot j$$

$$P^n = \left(1 \pm \frac{\sqrt{10}}{100} i\right)^n$$

Step 7

Choose one of the poles p, and make two sketches (on the same axis) of the real and imaginary parts of p^n when $k = 1$, $T = 0.1$ second, and $V = 0.1$ m/s. What does this plot tell you about the response of the system?

$$p = 1.0005 \cdot e^{0.0316j}$$

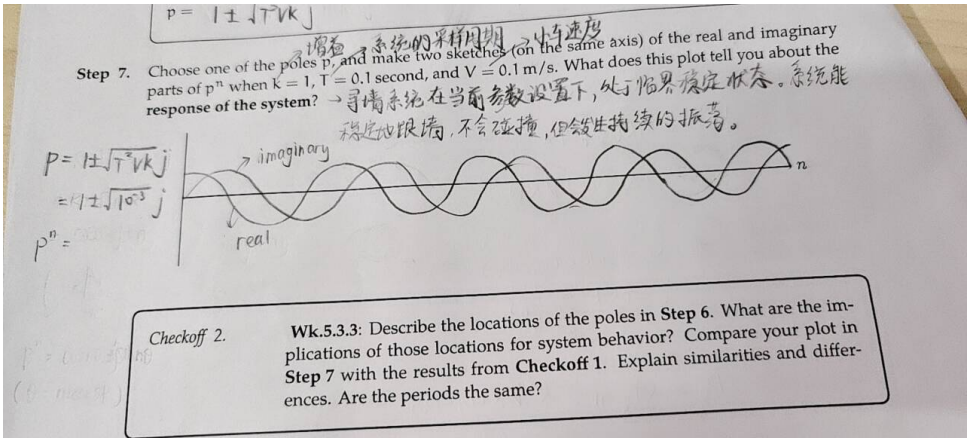


Fig. 8. Two sketches of the real and imaginary parts

Checkoff 2

Describe the locations of the poles in Step 6. What are the implications of those locations for system behavior? Compare your plot in Step 7 with the results from Checkoff 1. Explain similarities and differences. Are the periods the same?

When there are multiple poles,the characteristics of the system depend on the poles with larger absolute values, if its modulus is less than 1, the system oscillates and converges; if its modulus is greater than 1, the system oscillates and diverges.

Check Yourself 2

Use gains, delays, and adders to draw system diagrams representing the controller, plant 1, and plant 2 from the previous section

The system block diagrams are as follows:

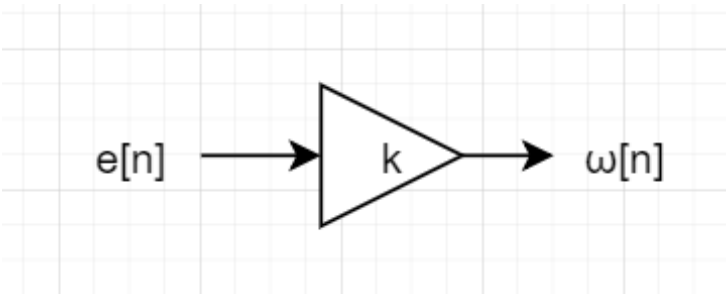


Fig. 9. Controller

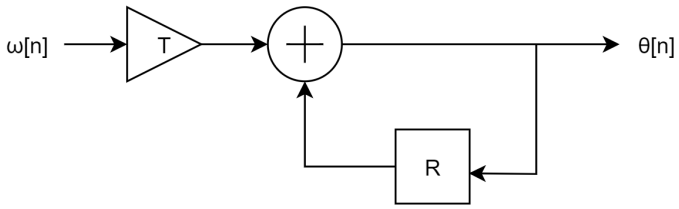


Fig. 10. Plant 1

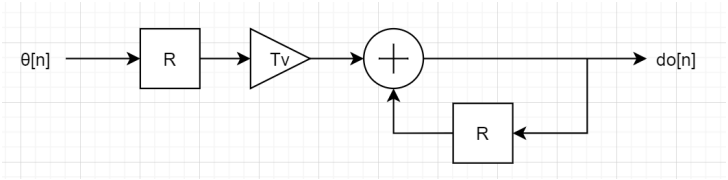


Fig. 11. Plant 2

Step 8/9

Edit the designLab05Work.py file to implement Python procedures called controller,plant1, and plant2 and write a Python procedure wallFollowerModel(k, T, V)

```
1       import lib601.sig as sig
2       import lib601.ts as ts
3       import lib601.poly as poly
4       import lib601.sf as sf
5
6       def controller(k):
7           return sf.Gain(k)
8       def plant1(T):
9           return sf.Cascade(sf.Gain(T),sf.FeedbackAdd(sf.Gain(1), sf.R()))
```

```
10      def plant2(T, v):
11          return sf.Cascade(sf.Cascade(sf.R(),sf.Gain(T*V)),sf.FeedbackAdd(sf.Gain(1),sf.R()))
12      def wallFollowerModel(k,T,V):
13          up = sf.Cascade(sf.Cascade(controller(k),plant1(T)),plant2(T, V))
14          return sf.Feedbacksubtract(up, sf.Gain(1))
```

Step 10

Use the dominantPole method of system model to determine the period that will result for T = 0.1 seconds, V = 0.1 m/s, and the values of k used in Checkoff 1.

When T = 0.1 seconds, V = 0.1 m/s, different k determine different poles and periods

When k = -20, the pole is (0.98+0.139j), the period is 14.4.

When k = -80, the pole is (0.93+0.262j), the period is 7.6.

When k = -100, the pole is (0.91+0.287j), the period is 6.9.

When $|k|$ getting bigger, the period is getting shorter

Checkoff 3

Show your results to a staff member. Explain similarities and differences between your results in this part and your results in Checkoff 1.

The calculation methods of the two are different, and one is an estimated value and the other is a measured value.

3 Summary

- We are exposed to a more complex system than before, and we are able to break down complex systems into several simple systems and analyze them step by step, which improved our abilities to analyze the system.
- We optimize the system parameters by programming and have a better understanding of the physical meaning of poles and the relationship between periods and poles.
- Due to the complex relationship between the cascade and feedback of the system, it is difficult for us to write the system functions in the form of general difference equations, therefore we learn to use operator expressions to express.